1998 International Computer Symposium
Workshop on Software Engineering and Database Systems
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

USING THESAURI FOR TERM-BASED COMPONENT RETRIEVAL: A DISTRIBUTED AND ADAPTIVE MODEL

Yuen-Chang Sun and Chin-Laung Lei

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan ROC
E-mail: sun@fractal.ee.ntu.edu.tw, lei@cc.ee.ntu.edu.tw

ABSTRACT

When performing a query against a software repository, a thesaurus mechanism can help the user overcome vocabulary and language barriers. Because of the difficulty and high cost of building and populating a thesaurus, however, such a mechanism is provided by few repositories. In this paper we propose a distributed and adaptive thesaurus model, which can reduce the cost of providing a thesaurus facility in a repository, make the thesaurus grow faster, and improve the performance of the query. The thesaurus architecture and the lookup algorithm are described formally, and an elementary analysis is given.

1 INTRODUCTION

The fast growth in services based on wide-area network technology has attracted the attention of the software reuse community [1, 2]. Traditional software repositories are designed as stand-alone services that do not communicate with each other. Currently more and more techniques and standards [3, 4] for building distributed and interoperating software repositories are emerging. By adopting a distributed architecture, software components can be shared among repositories to avoid wasting storage resources. Another advantage is that repositories can cooperate in executing a user query so that the search can be performed more thoroughly and more efficiently.

One important element in a repository system is the thesaurus. With a thesaurus, lexically unrelated terms can be correlated, improving the performance of a query. This is especially important for software repositories because in a rapidly growing field like software development an agreed vocabulary may not exist. Another reason for using a thesaurus is to overcome language barriers. A query written in one language can be issued to a repository of another language with the help of a thesaurus.

Roughly speaking the indexing and retrieval methods used by existing software repositories can be divided into six categories, namely enumerative methods [5, 6], keyword-based methods [5, 7], faceted methods [8, 9], attribute-value methods [10], free-text methods [11, 12], and specification-based methods [13, 14]. All these methods but the last category are *term-based* in the sense that the basic elements of their classification structure are *terms* like keywords, key phrases, attributes, and values. All the term-based repositories can benefit from the use of thesaurus, and some of them [5, 7, 8, 12] indeed include a thesaurus mechanism. For example, Prieto-Dìaz proposes a *conceptual graphs* mechanism, in which the correlation of two terms are defined as their distance in a weighted graph.

On the other hand, in the meantime distributed repositories are emerging, a thesaurus is still considered as part of the repository owning it and can not be shared. In this paper we introduce a distributed and adaptive thesaurus model, which has the following features and advantages not found in centralized thesaurus systems. First, thesauri can be shared. Repositories no longer have to develop and maintain their own thesauri, so the cost of populating a thesaurus, which is usually quite high, can be reduced. Second, thesauri can be co-maintained. A thesaurus can be populated in a cooperative fashion by parties who share the thesaurus, reducing the management cost each party must pay, and speeding up the expansion of the thesaurus. Both privileged managers and end users can contribute in this. Third, thesauri can be localized. Parties that access a publicly available repository can compose their queries with their own vocabularies or languages. This is done by letting them have their own local thesauri that can translate local terms into terms that are known by the repository. Finally, thesauri can be adapted. Thesaurus entries can be propagated, without explicit user intervention, in a group of thesauri that are used simultaneously. In the meantime the propagation can be controlled so that thesauri will not be overly adapted to erode locality.

2 OVERVIEW

In a software repository the purposes and functionality of a software component are described by an *index*.

When performing a query the correlation between the terms in the query and the terms in the component index must be found. This is done with the help of three thesaurus groups:

- the *local thesauri* are used by the client to find the synonyms of the terms in the query;
- the *foreign thesauri* are used by the client to translate the terms in the query and their synonyms into the language of the repository, if the languages of the client and the repository are different;
- the *remote thesauri* are used by the repository to find the synonyms of the terms and their synonyms, which may have been translated, in the query.

Both the local thesauri and the remote thesauri are "intra-lingual" thesauri in the sense that they correlate terms from the same language. In fact, a thesaurus can be both local and remote at the same time. On the other hand, a foreign thesaurus is "inter-lingual" in the sense that it correlates terms from different languages. In order to identify the language or languages involved, an intra-lingual thesaurus is identified by a *language code*, and in inter-lingual thesaurus is identified by two language codes.

With the thesauri the query procedure is divided into eight steps. They are illustrated in Figure 1 and briefly described below. Details can be found in Section 3 and Section 4. (1) A query is issued. (2) The terms in the query are extracted by the client system and are sent to the local thesauri. (3) A sub-thesaurus containing all the thesaurus entries that involve the terms in the query is returned from the local thesauri to the client. This sub-thesaurus defines all the synonyms of the terms in the query. (4) The terms in the query, together with their synonyms found in the previous step, are sent to the foreign thesauri. (5) Similar to the third step, a foreign sub-thesaurus is constructed and sent back. It defines the translations of the query terms and their synonyms. (6) Both the two sub-thesauri, plus the query itself, are sent to the repository. (7) The translations of the query terms and their synonyms are sent to the remote thesauri. (8) Another sub-thesaurus containing the synonyms of the translated terms are returned from the remote thesaurus to the repository. Finally the three sub-thesauri are combined into one, which is used to find the correlation between the query terms and the index terms later by the query engine. Note that in the case that the languages of the client and the repository are the same, the fourth and fifth steps are skipped.

It may be argued that transmitting whole sub-thesauri between sites is wasteful because many of the entries will turn out to be useless. However, our approach can avoid sending many small messages on the network, which can more easily decrease network performance.

Thesauri are stand-alone servers that can be shared by multiple clients and repositories. They can be configured as purely private to a single user, or owned by a small group, or available to the public. If necessary they can be protected by password or other methods to restrict access. Narrowly available thesauri are used for dealing with local vocabularies or languages. For example, clients or repositories can establish private thesauri for handling abbreviations and acronyms that are uncommon to other parties.

By using multiple thesaurus simultaneously, thesaurus entries can be propagated among thesaurus servers. After a lookup to a group of thesauri is completed, the results are combined and fed back to the involved thesauri. Then the thesaurus entries can be circulated within the group, and if one of the thesauri is a member of another thesaurus group, the thesaurus entries can further be propagated. Thesaurus servers do not accept term definitions blindly, though. They have their own acceptance policies that can prevent them from being overly populated or attacked. Details about this are given in Section 5.

3 THESAURI

A thesaurus θ is a function that, given two terms w and w', returns their correlation value $\theta(w, w')$ such that $0 \leq \theta(w, w') \leq 1$. The larger the function value, the higher the correlation between the two terms. A function value 1 means equivalence in meanings, and a 0 means irrelevance.

There are two kinds of thesauri, namely intra-lingual thesauri and inter-lingual thesauri. Intra-lingual thesauri have the properties that $\theta(w, w') = \theta(w', w)$ for any w and w', and $\theta(w, w) = 1$ for any w. The language of an intra-lingual thesaurus can be explicitly specified by the notation θ^{λ}, where λ is the language code. The languages of an inter-lingual thesaurus can be specified similarly by $\theta^{\lambda,\lambda'}$. Inter-lingual thesauri do not have the above two properties because in $\theta^{\lambda,\lambda'}(w, w')$ the terms w and w' must be from different languages, λ and λ' respectively, so they can neither be exchanged nor be identical.

Two special terms, the D/C (don't care) symbol "$*$" and the N/A (not applicable) symbol "$/$", are defined for special purposes. The D/C symbol is equivalent to any term, that is, $\theta(*, w) = \theta(w, *) = 1$ for any w, and the N/A symbol is irrelevant to any term, that is, $\theta(/, w) = \theta(w, /) = 0$ for any w. $\theta(*, /)$ and $\theta(/, *)$ are defined as 0.

Pattern matching can be performed by including wildcards or even regular expressions in a term. In this case the term is called an *improper term*. Thus there are two kinds of identity: "$w = w'$" denotes that w and w' are lexically identical, and "$w \approx w'$" denotes that w, which may or may not be improper, is pattern-matched with w'. Note that an exact match implies a pattern match. Also note that patterm matches are not

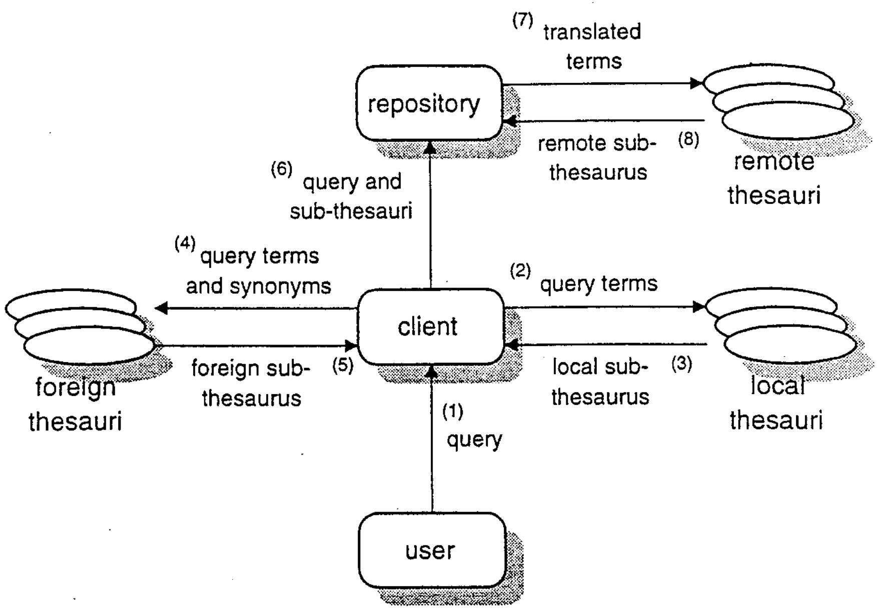

Figure 1: Thesaurus configuration overview.

symmetric because only the first term can be improper.

A thesaurus is also a set of thesaurus entries, which are triples in the form (w, w', r), where r is the correlation between the terms w and w'. The entry set of a thesaurus θ is also denoted by θ. Since a thesaurus entry set forms a function from w and w' to r, there are some restrictions on which entries can coexist in an entry set. For an intra-lingual thesaurus θ, if $(w, w', r) \in \theta$, then θ cannot contain (w, w', r') for any $r' \neq r$, or (w', w, r'') for any r''. In fact, for brevity the semantics of the "$\in$" operator is extended for intra-lingual thesauri in this paper: $(w, w', r) \in \theta$ means θ contains either (w, w', r) or (w', w, r). For an inter-lingual thesaurus θ, if $(w, w', r) \in \theta$, then θ cannot contain (w, w', r') for any $r' \neq r$. Besides these restrictions, it is not necessary to store all possible combinations in a thesaurus entry set. Entries with identical terms should not appear in a thesaurus entry set, and special terms such as "$*$", "$/$" and improper terms should not appear in a thesaurus entry. In addition, the number of entries involving a specific term should be limited so that only a number of entries with the largest r are kept. In practice keeping at most 10 entries for a term is sufficient. This not only can save storage space but also can improve efficiency.

Before the query system can try to match a query with a component index, a thesaurus combining the parts of the local, foreign and remote thesauri that involve the query terms must be constructed. The lookup procedure is explained in Section 4 in detail. Before that several thesaurus operations are defined in the following. First, the subset of a thesaurus θ involving a given term w is

$$\theta(w) = \{(\tilde{w}, w', r) \mid w \approx \tilde{w} \text{ and } (\tilde{w}, w', r) \in \theta\}.$$

```
function $\theta(w, w')$ is
    if $\theta$ is inter-lingual and $w \approx w'$ then
        return 1
    else if $w = /$ or $w' = /$ then
        return 0
    else if $w = *$ or $w' = *$ then
        return 1
    else if $\theta(w)$ is not empty then
        return max $\{r \mid (\tilde{w}, w', r) \in \theta(w)\}$
    else
        return 0
end
```

Figure 2: The Term Correlation Algorithm

Note that this defines a new thesaurus, a sub-thesaurus of θ. The second operation is an extension to the first one, which works on a set Ω of terms:

$$\theta(\Omega) = \biguplus_{w \in \Omega} \theta(w)$$

where "$\uplus$" is the *thesaurus union operator*. The union $\theta_1 \uplus \theta_2$ is defined as the set union of θ_1 and θ_2 so that only the entry with a larger r is included in the resulted set if entries for w and w' can be found in both θ_1 and θ_2. Finally, the correlation between two terms w and w' with respect to a thesaurus θ, which may be a combination of other thesauri, is defined by the algorithm shown in Figure 2.

4 THESAURUS LOOKUP

For performing a query, several pieces of information concerning the languages and thesauri involved are essential. First, the languages used by the repository and the client must be known. Both of them must be using at least one language, and mixing multiple languages in a query or a component index is not unusual. It is possible that the repository and the client are using different languages. Second, the client should specify a set of local thesauri and a set of foreign thesauri. It is beneficial to use more than one thesaurus simultaneously in those two sets for locality and adaptation reasons. It is possible that the thesauri in a set are working on different languages. For example, if the user is constantly mixing two languages in queries, intra-lingual thesauri working on both those languages should be included in the local thesaurus set. Also, if the user is querying against repositories using different languages, inter-lingual thesauri working on different pairs of languages should be included in the foreign thesaurus set. The query system can choose different groups of thesauri according to the languages used by the repositories. Third, the server should specify a set of remote thesauri. Finally, each thesaurus must be specified as either read/write or read-only. These two types of working modes place different acceptance policies over feedbacks.

4.1 The Simple Case

The case that both the client and the repository are using one language, possibly different ones, is discussed first. Suppose the language used by the client and the repository are λ and λ', respectively. All the local thesauri that works on λ form a *thesaurus group* $\Theta_L = \{\theta_{L_1}, \dots, \theta_{L_s}\}$. Similarly, the foreign thesaurus group is $\Theta_F = \{\theta_{F_1}, \dots, \theta_{F_t}\}$, in which all the thesauri work on λ and λ', and the remote thesaurus group is $\Theta_R = \{\theta_{R_1}, \dots, \theta_{R_u}\}$, which works on λ'. The terms in the query are collected in the set $\Omega = \{w_1, \dots, w_k\}$.

The first step of lookup is sending Ω to all the thesauri in Θ_L and combining the results. Let $\theta_\alpha = \Theta_L(\Omega)$, where the lookup operation by a thesaurus group Θ on a term set Ω is defined as

$$\Theta(\Omega) = \biguplus_{\theta \in \Theta} \theta(\Omega).$$

θ_α defines the correlation between the terms in Ω and their synonyms. Those synonyms are then extracted and unioned with Ω:

$$\Omega_\alpha = \vec{\theta}_\alpha(\Omega) \cup \Omega.$$

In the above the *right term set* of a thesaurus θ with respect to a term set Ω is defined as

$$\vec{\theta}(\Omega) = \{w' \mid \exists(\tilde{w}, w', r) \in \theta \text{ such that } w \approx \tilde{w} \text{ for some } w \in \Omega\}.$$

The next step depends on whether $\lambda = \lambda'$ or not. If $\lambda \neq \lambda'$, the foreign thesaurus group is used for translating the terms in Ω_α to $\theta_\beta = \Theta_F(\Omega_\alpha)$, and then the translated terms are extracted as $\Omega_\beta = \vec{\theta}_\beta(\Omega_\alpha)$. Note that this time the right term set $\vec{\theta}_\beta(\Omega_\alpha)$ is not unioned with Ω_α or Ω because they are in different languages. If $\lambda = \lambda'$, θ_β is not constructed and Ω_β is simply Ω_α.

The above actions are all initiated by the client. After that, θ_α, θ_β (if $\lambda \neq \lambda'$) and Ω_β, together with the user query, are sent to the repository. The remote thesaurus group is then used for finding synonyms in λ' as $\theta_\gamma = \Theta_R(\Omega_\beta)$. Finally the combined thesaurus, which will be consulted as the query is actually performed by the repository, is established by crossing up all the intermediate thesauri as $\theta = \theta_\alpha \times \theta_\beta \times \theta_\gamma$ if $\lambda \neq \lambda'$, or $\theta = \theta_\alpha \times \theta_\gamma$ otherwise. The cross operation is defined so that the r values of the entries in the result are maximazed.

4.2 The General Case

The case that the client and/or the repository are using multiple languages is much more complicated, but the basic algorithm is the same. Suppose the client language set is $\{\lambda_1, \dots, \lambda_n\}$, the repository language set is $\{\lambda'_1, \dots, \lambda'_m\}$, and the query term set is still $\Omega = \{w_1, \dots, w_k\}$. This time there are multiple local thesaurus groups $\Theta_L^{\lambda_1}, \dots, \Theta_L^{\lambda_n}$ working on each local language λ_i, $1 \leq i \leq n$, multiple foreign thesaurus groups $\Theta_F^{\lambda_1,\lambda'_1}, \dots, \Theta_F^{\lambda_n,\lambda'_m}$ working on each pair of languages λ_i and λ'_j, $1 \leq i \leq n$, $1 \leq j \leq m$ and $\lambda_i \neq \lambda'_j$, and multiple remote thesaurus groups $\Theta_R^{\lambda'_1}, \dots, \Theta_R^{\lambda'_m}$ working on each remote language λ'_j, $1 \leq j \leq m$. It seems that there are too many foreign thesaurus groups involved: in the worst case there are nm groups. However, in practice it is rare to mix more than two languages, so usually at most four groups are needed. Besides, it is likely that the client and the repository are using some languages in common, and the number of foreign thesaurus involved can further be reduced. For example, if the client and the repository are both using Chinese and English—a common practice of Chinese natives—only one Chinese-English inter-lingual thesaurus group suffices. We do not need two (Chinese-English and English-Chinese) because an inter-lingual thesaurus can be used backwardly.

The multiple-language case is a natural generalization of the simple case. First the query terms are sent to each of the local thesaurus groups: $\theta_\alpha^{\lambda_i} = \Theta_L^{\lambda_i}(\Omega)$ for $1 \leq i \leq n$. All the terms, no matter what languages they are in, are sent to each of the thesaurus groups because there is no good way to tell which term is in which language in general; the absence of a term in a thesaurus θ^λ does not imply that the term is not in λ. In certain cases, however, it is easy to tell which term is in which language, and in those cases minor modifications can be made to the algorithm proposed here to make use of this knowledge.

The next step is extracting the synonyms out of the thesauri created in the first step as $\Omega_\alpha^{\lambda_i} = \vec{\theta}_\alpha^{\lambda_i}(\Omega) \cup \Omega$, and then sending them to the foreign thesaurus groups to get $\theta_\beta^{\lambda_i,\lambda'_j} = \Theta_F^{\lambda_i,\lambda'_j}(\Omega_\alpha^{\lambda_i})$ for $1 \le i \le n$, $1 \le j \le m$, $\lambda_i \ne \lambda'_j$. The translated terms in each of the repository languages are extracted as

$$\Omega_\beta^{\lambda'_j} = \left(\bigcup_{1 \le i \le n}^{\lambda_i \ne \lambda'_j} \vec{\theta}_\beta^{\lambda_i,\lambda'_j}(\Omega_\alpha^{\lambda_i}) \right) \cup \Omega_\alpha^{\lambda'_j}$$

if $\lambda'_j \in \{\lambda_1, \ldots, \lambda_n\}$, or as

$$\Omega_\beta^{\lambda'_j} = \left(\bigcup_{1 \le i \le n}^{\lambda_i \ne \lambda'_j} \vec{\theta}_\beta^{\lambda_i,\lambda'_j}(\Omega_\alpha^{\lambda_i}) \right)$$

otherwise, for $1 \le j \le m$. All the thesauri and term sets are then sent to the repository, which consults the remote thesaurus groups for synonyms as $\theta_\gamma^{\lambda'_j} = \Theta_R^{\lambda'_j}(\Omega_\beta^{\lambda'_j})$ for $1 \le j \le m$, and finally establishes the combined thesaurus $\theta =$

$$\left(\biguplus_{\lambda_i \ne \lambda'_j} (\theta_\alpha^{\lambda_i} \times \theta_\beta^{\lambda_i,\lambda'_j} \times \theta_\gamma^{\lambda'_j}) \right) \uplus \left(\biguplus_{\lambda_i = \lambda'_j} (\theta_\alpha^{\lambda_i} \times \theta_\gamma^{\lambda'_j}) \right).$$

Note that θ is a combination of thesauri working on different languages. This does not cause problems because it is used only internally in a single query session and exists only temporarily.

4.3 Efficiency Considerations

A major concern about the lookup algorithm is its performance, which can roughly be measured by calculating the amount of data transferred from the client to the repository. A worst-case analysis is given in the following.

Suppose the client language set is $\{\lambda_1, \lambda_2\}$, the repository language set is $\{\lambda'_1, \lambda'_2\}$, and these two sets are disjoint. Eight terms are contained in the query term set, and there are two thesauri in each thesaurus group. Remembering that for any specific term at most ten entries are kept in a thesaurus, we have:

$$\begin{aligned}
|\theta_\alpha^{\lambda_i}| &\le |\theta_{L_1}^{\lambda_i}| \times 2 \le (8 \times 10) \times 2 = 160, \\
|\Omega_\alpha^{\lambda_i}| &\le |\vec{\theta}_\alpha^{\lambda_i}(\Omega)| + |\Omega| \le 160 + 8 = 168, \\
|\theta_\beta^{\lambda_i,\lambda'_j}| &\le |\theta_{F_1}^{\lambda_i,\lambda'_j}(\Omega_\alpha^{\lambda_i})| \times 2 \le (168 \times 10) \times 2 = 3360, \\
|\Omega_\beta^{\lambda'_j}| &\le |\vec{\theta}_\beta^{\lambda_i,\lambda'_j}(\Omega_\alpha^{\lambda'_j})| \times 2 \le 3360 \times 2 = 6720.
\end{aligned}$$

What have to be sent to the repository are the thesauri $\theta_\alpha^{\lambda_i}$, $1 \le i \le 2$, and $\theta_\beta^{\lambda_i,\lambda'_j}$, $1 \le i, j \le 2$, and the term sets $\Omega_\beta^{\lambda'_j}$, $1 \le j \le 2$. With encoding a term can be represented as two bytes, and so is a correlation value, so the total size is $160 \times 2 + 3360 \times 4 + 6720 \times 2 \approx$ 110K. This seems making a big deal out of a task of secondary importance, but the actual situation is not so bad. First, the analysis here is quite pessimistic. For example, it assumes 20 synonyms can be found for a term in the query, 20 translations can be found for each of those synonyms, and all those 400 translations for that query term are distinct, which is extremely unlikely. It also assumes that the client and the repository are not using a language in common, which is usually not the case. In practice we can expect the number of distinct translations of a term to be no more than 20, and at least one language is used in common by both parties, reducing the amount of transmitted data by a factor of at least 40. Second, the algorithm can be optimized for efficiency by crossing θ_α and θ_β before sending them to the repository. With the "ten entries for a term" restriction, the combined thesaurus will be about the same size as θ_α is, further reducing the amount of data to be transferred, with the price of decreased precision. Finally, even the worst case does hold, with the next generation wide-area network transmitting such amount of data will not be impractical.

5 THESAURUS ADAPTATION

There is no difference in functionality between a thesaurus group and a single thesaurus from the viewpoint of the lookup algorithm. There is, however, an additional important task for the thesauri in a thesaurus group, that is, adapting themselves to new thesaurus entries.

Upon receiving a term set $\Omega = \{w_1, \ldots, w_k\}$ each thesaurus in the thesaurus group $\Theta = \{\theta_1, \ldots, \theta_n\}$ works on Ω and returns a sub-thesaurus. The resulted sub-thesauri are unioned together to form the output of Θ:

$$\Theta(\Omega) = \biguplus_{1 \le i \le n} \theta_i(\Omega).$$

The action performed by the thesaurus group for adaptation is simply comparing $\Theta(\Omega)$ to each $\theta_i(\Omega)$, $1 \le i \le n$, and feeding the difference $\overline{\theta}_i = \Theta(\Omega) - \theta_i(\Omega)$ back to it. The difference $\theta_1 - \theta_2$ of two thesauri θ_1 and θ_2 is defined as

$$\{(w, w', r) \in \theta_1 \mid (w, w', r') \notin \theta_2 \text{ for any } r'\}.$$

How θ_i makes use of $\overline{\theta}_i$ depends on whether θ_i is opened in read/write mode or read-only mode. Before discussing this the *citizenship* of thesaurus entries must be introduced. An entry in a thesaurus can be either

- a *citizen*, which can stay in the thesaurus as a citizen permanently until explicitly removed by the manager, and is *visible* in that it is effective during a lookup, or

- a *resident*, which is visible but is not permanent, meaning that the system can automatically downgrade it to the lower class, or

- an *applicant*, which is neither visible nor permanent, meaning that it is not effective during a lookup, and the system can automatically throw it out of the thesaurus.

When a read/write thesaurus receives a feedback, the entries are included as residents. Since residents are visible, they become effective immediately. This usually occurs when the user is using a public thesaurus in read-only mode and a private thesaurus in read/write mode. The private thesaurus can grow by absorbing the richer collection in the public one. Later if the public thesaurus becomes unreachable, or is disconnected by the user for some reason, the previously fetched entries will still be "cached" in the private thesaurus. The number of residents in a thesaurus is limited, and the oldest residents will be downgraded to the applicant class when this limit is exceeded. On the other hand, the manager can periodically survey the residents and upgrade some of them to the citizen class. Note that no entry can enter the citizen class without an explicit permission from the manager.

When a read-only thesaurus receives a feedback, the entries are included as applicants. Entries in this class are not visible. Read-only thesauri are usually shared by unrelated users, so it is inappropriate to make feedback entries visible immediately. On the other hand, if an entry has been recommended by several independent parties, it may be feasible to upgrade that entry to the resident class, making it visible. Like the resident class, the number of applicant entries is also limited, and the oldest entries will be discarded when this limit is exceeded.

In summary, the feedback acceptance policy is as the following. First, read/write thesauri accept feedback entries as residents, and read-only thesauri accept them as applicants. Second, if the correlation between two terms is fed back as applicants by u independent parties within p days, the entry can be upgraded to the resident class. It is likely that those parties suggest different correlation values, and the correlation value in the upgraded entry will be their average. The repository must make sure that those parties are independent to avoid being attacked. This can be done by checking their network addresses. The recommendations must be received within p days so that aged recommendations will not accumulate. The parameters u and p are determined by the thesaurus manager. Third, the number of residents and the number of applicants are limited, and the oldest entries will be downgraded or discarded if the limits are exceeded. Finally, only the thesaurus manager has the privilege to upgrade an entry to the citizen class.

6 CONCLUSIONS

In this paper we propose a distributed and adaptive thesaurus model for term-based component retrieval. Traditionally, it is considered laborious and time-consuming to construct a thesaurus, and few repository are investigating in it. With our model, the cost of developing and maintaining thesauri can be shared by thesaurus users, making it affordable for even small repositories to use thesaurus facilities. The frustration of finding an alient vocabulary or in the repository being queried can be alleviated, and regional users can benefit from being able to issue queries in their native languages.

Our model is adaptive in that thesaurus entries can automatically be propagated from one thesaurus server to another. This propagation is controlled by a sophisticated acceptance policy, reserving locality, and avoiding malicious actions.

The model is described formally in this paper, and can be applied widely on various systems. The amount of data transmitted during a lookup is analyzed and is found acceptable. Although adopting such a thesaurus mechanism in a repository system needs extra work, we believe that the expected benefit is worth the effort paid.

REFERENCES

[1] G. Arango, "Software Reusability and the Internet," *ACM Symposium on Software Reusability*, pp. 22–23, Seattle, WA, USA, April 1995.

[2] S. Browne and J. Moore, "Reuse Library Interoperability and the World Wide Web," *ACM Symposium on Software Reusability*, pp. 182–189, MA, USA, April 1997.

[3] S. Browne, J. Dongarra, S. Green, K. Moore, T. Pepin, T. Rowan and R. Wade, "Location-Independent Naming for Virtual Distributed Software Repositories," *ACM Symposium on Software Reusability*, pp. 179–185, Seattle, WA, USA, April 1995.

[4] *IEEE Standard for Information Technology—Software Reuse—Data Model for Reuse Library Interoperability: Basic Interoperability Data Model (BIDM)*. IEEE Standard 1420.1, 1995.

[5] B. A. Burton, R. W. Aragon, S. A. Bailey, K. D. Koehler and L. A. Mayes, "The Reusable Software Library," *IEEE Software*, vol. 4, no. 7, pp. 25–33, 1987.

[6] *IMSL Math/Library User's Manual*. Houston, Texas, 1987.

[7] S. P. Arnold and S. L. Stepoway, "The Reuse System: Cataloging and Retrieval of Reusable Software," *Proceedings of COMPCON S'87*, pp. 376–379, 1987.

[8] R. Prieto-Dìaz and P. Freeman, "Classifying Software for Reusability," *IEEE Software*, vol. 4, no. 1, pp. 6–16, 1987.

[9] J. S. Poulin, "Populating Software Repositories: Incentives and Domain-Specific Software," *Journal of Systems Software*, vol. 30, pp. 187–199, 1995.

[10] P. Devanbu, R. J. Brachman, P. G. Selfridge and B. W. Ballard, "LaSSIE: A Knowledge-Based Software Information System," *Communications of the ACM*, vol. 34, no. 5, pp. 34–49, 1991.

[11] Y. S. Maarek, D. M. Berry and G. E. Kaiser, "An Information Retrieval Approach for Automatically Constructing Software Libraries," *IEEE Transactions on Software Engineering*, vol. 17, no. 8, pp. 800–813, 1991.

[12] W. B. Frakes and B. A. Nejmeh, "An Information System for Software Reuse," *Proceedings of the Tenth Minnowbrook Workshop on Software Reuse*, 1987.

[13] P. S. Chen, R. Hennicker and M. Jarke, "On the Retrieval of Reusable Components," *Selected Papers from the Second International Workshop on Software, Reusability Advances in Software*, ReuseLucca, Italy, 1993.

[14] A. Mili, R. Mili and R. Mittermeir, "Storing and Retrieving Software Components: A Refinement-based Approach," *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy, 1994.